# ALGORITMOS PARA LA BÚSQUEDA DE TEXTO

Miguel Ángel Torres Cardona
*Universidad Distrital "Francisco José de Caldas"*
Patricia Páez Cárdenas
*Universidad Distrital "Francisco José de Caldas"*
Carmen Alicia Martínez Rivera
*Universidad Distrital "Francisco José de Caldas"*
Jorge Enrique Rodríguez Rodríguez,
*Universidad Distrital "Francisco José de Caldas"*
*Bogotá (Colombia)*

Resumen - Se expone los conceptos y algoritmos relacionados con la búsqueda de información. En primer lugar, se habla acerca de los modelos de recuperación de información en orden de eficiencia que corresponden a los modelos Booleanos, Probabilístico y Vectoriales. En segundo lugar, se mencionan, algunas tecnologías que intervienen en el análisis de texto. Finalmente, se indican los algoritmos de búsqueda de texto más utilizados.

Palabras clave: indexación, recuperación de información, sintaxis, morfología, palabra, indización.

## I. INTRODUCCIÓN

La Recuperación de Información (RI) es la ciencia que estudia las metodologías para la búsqueda de información y/o datos en documentos digitales, esto incluye el desarrollo de algoritmos, pre procesamiento de datos y estudio del problema a tratar, teniendo en cuenta este concepto se han desarrollado aplicaciones o herramientas encargadas del análisis sintáctico y semántico de las palabras, herramientas como: los analizadores sintácticos, morfológicos y desambiguadores, cada una con una función específica. Estas herramientas han sido desarrolladas mediante un conjunto de algoritmos que utilizan una lógica específica para la búsqueda de información y análisis de texto; dentro de los algoritmos están: algoritmo Knuth-Morris-Pratt (Kmp), algoritmo Boyer-Moore, shift-or, arboles de sufijos, el algoritmo de caracteres lejanos, algoritmo de q-gramas y el algoritmo Chang-Marr. El presente artículo aborda los temas concernientes al análisis, búsqueda y clasificación de texto en cuanto a modelos, técnicas y algoritmos propuestos, con el fin de encontrar la mejor alternativa para desarrollo de una herramienta de software que permita la recuperación de texto a partir de un contexto determinado.

## II. MODELOS DE RECUPERACIÓN DE INFORMACIÓN

El punto central en este artículo es la Recuperación de Información, cuya definición, según [1] es: *"la recuperación de información (IR) se ocupa de encontrar* material (normalmente documentos) de una naturaleza no estructurada (normalmente texto) que se encuentra en grandes colecciones (normalmente almacenada en forma digital) y satisface una necesidad de información.". Las tareas más representativas que intervienen en el proceso de recuperación de información (RI) se ilustran en la Fig.1. Como se observa existe una necesidad de información y documentos que pueden contener información valiosa sobre la consulta que se desea. Tanto la consulta como el documento son representados de una manera lógica para hacer una comparación. El usuario recibe los datos más acercados a su consulta. En el mejor de los casos, esta consulta no resulta siendo el final del proceso, puesto que puede ser utilizado para retroalimentar el algoritmo por medio de calificación del usuario con respecto al documento encontrado por la herramienta.[15] Los métodos utilizados para representar tanto documentos como consultas en sistemas de RI y comparar su similitud son los modelos de recuperación.

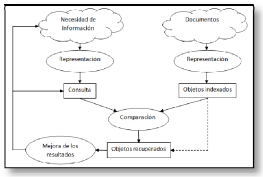


***Fig.1 Tareas en la recuperación de información [2]***

### *2.1 Modelo Booleano*

Este modelo está basado en la teoría de conjuntos y el álgebra de Boole. Las consultas se formulan como expresiones lógicas que combinan operadores de álgebra booleana. Serán relevantes los documentos que contengan los términos claves que se indican en la consulta. La ventaja es su sencillez, su desventaja es la dificultad para formular exactamente las necesidades de información de los usuarios con algebra de Boole y

términos clave; además, solo obtiene documentos que se ajustan a la consulta. [2]

### *2.2 Modelo Probabilístico*

Propuesto por Robertson y Spark-Jones, el objetivo es verificar la probabilidad de existencia o asociación de relevancia entre una consulta y un número determinado de documentos. A partir de una expresión de consulta se puede dividir una colección de

### *2.3 Modelo Vectorial*

Este modelo construye y asigna un vector tanto a la consulta *t* como al documento *d,* cada uno de estos vectores almacenan un número de palabras que son analizadas e indexadas con un número que determina la importancia de la palabra (peso) para la consulta. El vector de consulta es comparado con los vectores de los demás documentos de la colección de archivos. En teoría, los vectores que posean palabras o términos iguales/similares, tendrán mayor importancia y se mostrarán como primeros en la lista de más relevantes. Para observar esa similitud de relevancia entre *t* y *d*, se usan modelos dentro de los que se destaca el modelo del coseno. En este caso, a cada palabra debe asignársele un peso entre 0 y 1, siendo 1 el de mayor relevancia. [25]

A) Apache Lucene: Lucene es una librería escrita completamente en Java, permite el desarrollo de motores de búsqueda textuales de alto rendimiento. Lucene emplea una representación de tipo objeto de los documentos indexados. Implementa campos (nombre, valor) e información sobre, si esta palabra es almacenada o indexada dentro de los criterios de búsqueda. Los valores de los campos se establecen empleando analizadores, los cuales implementan técnicas de limpieza que se emplean habitualmente en sistemas de RI para reducir el número de palabras indexadas y mejorar el rendimiento del índice. En la etapa de indexación textual del flujo de trabajo el sistema construye un índice. Cada una de las abstracciones de documentos construidas en la etapa anterior se inserta en el índice textual. El identificador de documento se almacena pero no se indexa en el índice textual y cada campo marcado como indexable en el índice textual o en ambos índices se indexa (dividido en sus componentes léxicos). [19]

B) Sistema Smart: Este método asigna a cada término de la consulta un peso que puede ser un valor positivo. Los coeficientes binarios más utilizados como medida de similitud entre una consulta y los documentos en los sistemas de RI son producto escalar, coeficiente del coseno, coeficiente de Dice o coeficiente de Jaccard. [4]

*N* documentos en cuatro subconjuntos distintos: *REL* conjunto de documentos relevantes, *REC* conjunto de documentos recuperados, *RR* conjunto de documentos relevantes recuperados y *NN* conjunto de documentos no relevantes no recuperados. [3] El resultado ideal de este modelo es cuando *REL* es igual a *REC*, el problema que presenta este algoritmo es en el tiempo de respuesta efectiva, dado que generalmente el usuario debe suministrar una evaluación del resultado y retroalimentar el algoritmo, lo cual conlleva a un tiempo de interacción prolongado.

## III. ALGORITMOS DE BÚSQUEDA EN TEXTO

Los algoritmos de búsqueda en texto se dividen en dos grupos: - algoritmos de búsqueda secuencial, los cuales buscan un patrón específico en el texto, y - algoritmos de búsqueda en texto permitiendo errores, en este caso se tiene una tasa de error en el momento de la búsqueda pero permite hacer una búsqueda aproximada de la consulta.

### *3.1 Algoritmos de Búsqueda Secuencial*

La función de los algoritmos de búsqueda en texto secuencial, es encontrar la ocurrencia de un patrón exacto en el texto, entendiendo como patrón una palabra o serie de palabras de interés para el usuario. Estos algoritmos a su vez se sub-dividen en dos categorías, algoritmos de corrimiento y algoritmos Boyer – Moore.

A) Algoritmo Knuth-Morris-Pratt (KMP): El algoritmo de fuerza bruta, realiza una comparación simple, puesto que compara las palabras del texto con el patrón de izquierda a derecha; cuando no encuentra ocurrencia entre la consulta y el texto genera un desplazamiento del patrón una posición hacia la derecha y vuelve a realizar la comparación. Sin embargo, esto puede o no ser adecuado, en el sentido que los primeros *j-1* caracteres de *X* pueden o no coincidir, es decir; que se presente una coincidencia en las primeras letras entre las cadenas comparadas, pero que al finalizar no sean iguales. Para ejemplificar el funcionamiento del algoritmo KMP se supone *X* como una parte del patrón donde hay ocurrencia con el texto *Y*; además, el largo de *X* sería *j.* La observación clave que realiza el algoritmo KMP es que *X* es igual a *Y*, por lo cual la búsqueda se puede basar en el patrón; en el momento de realizar la búsqueda, el algoritmo hace una comparación y si no encuentra concordancia decide hacer un salto del patrón y deslizarlo en una posición, si no funciona, se puede intentar deslizarlo en 2, 3,..., hasta *j* posiciones. Por su parte, *f(j)* es la longitud del mayor prefijo de *X*, que además es sufijo de *X*. Note que *j = 1* es un caso especial, puesto que si hay una discrepancia en buscar la primera posición del patrón *b1* , el patrón se desliza en una posición, lo cual se refiere a que el corrimiento de comparación entre el patrón y el texto depende de la longitud *j* del patrón.

[11] El patrón hace una comparación con el texto, empezando desde la posición *bj=1*; si compara la siguiente posición, *bj+1*, y encuentre una discrepancia, el patrón hace un corrimiento, de manera que la función que halla la cantidad de caracteres que se deben correr *bf(j)*, correrá hasta la posición donde se encontró la última coincidencia, es decir *bj*, y se compara nuevamente.[17]

B) Algoritmo Boyer-Moore: Hasta el momento, los algoritmos de búsqueda en texto siempre comparan el patrón con el texto de izquierda a derecha. Sin embargo, suponga que la comparación ahora se realiza de derecha a izquierda: si hay una discrepancia en el último carácter del patrón y el carácter del texto no aparece en todo el patrón, entonces éste se puede deslizar m posiciones sin realizar ninguna comparación extra. El método descrito es la base del algoritmo Boyer-Moore, del cual se describirán dos variantes: Horspool y Sunday.

- Boyer-Moore-Horspool (BMH): examina los caracteres del patrón con el texto iniciando con el carácter más a la derecha, en caso de una no ocurrencia o una ocurrencia parcial del patrón usa una función previa para hacer una comparación más a la izquierda en lo que se denomina salto del mal carácter. El salto del mal carácter consiste en alinear cada carácter del alfabeto Σ con la ocurrencia más a la derecha en x[0,…, m-2]. Si el carácter no ocurre en el patrón *X*, la no ocurrencia de *X* puede incluir el carácter, y alinearlo en el lado más izquierdo del patrón. Esta operación (usada en el algoritmo BM) no es muy eficiente para alfabetos pequeños, pero cuando el alfabeto es grande comparado con la longitud del patrón llega a ser muy útil [5]. Este algoritmo, en comparación con el algoritmo de fuerza bruta, permite realizar la búsqueda en un lapso de tiempo menor, esto debido a que el corrimiento cuando no hay concordancia está entre uno y *j* caracteres, siendo *j* la longitud del patrón. Se puede demostrar que el tiempo promedio que toma el algoritmo BMH es el dado en la ec. 1, donde c es el tamaño del alfabeto (*c*<<*n*). En el peor caso, BMH tiene el mismo tiempo de ejecución que el algoritmo de fuerza bruta.

$O\left(n\left(\frac{1}{m}+\frac{1}{2c}\right)\right)$ ***Ec. 1. Tiempo promedio del algoritmo BMH***

- Boyer-Moore-Sunday (BMS): El algoritmo BMS desliza el patrón basado en el símbolo del texto que corresponde a la posición del último carácter del patrón. Este siempre se desliza al menos una posición si se encuentra una discrepancia con el texto.

C) Shift-or[6]: Este algoritmo trabaja con un conjunto de caracteres, un alfabeto y utiliza una máquina de estados. El alfabeto lo forma todos los caracteres diferentes del texto y del patrón. Los estados son la comparación de las diferentes partes del patrón con diferentes partes del texto, en donde se utilizan dos operaciones: corrimiento (SHIFT), paso de un estado a otro, y la operación lógica OR, para calcular el siguiente estado. Para la búsqueda se toma el estado actual, se le hace corrimiento a la izquierda de *1* bit y se le aplica la operación OR con el patrón correspondiente al carácter actual. Finalmente, se corre un carácter hacia la derecha [7].

D) Árboles de sufijos [8]: Un árbol de sufijos es una estructura de datos que sirve para almacenar una cadena de caracteres con "información pre-procesada" sobre su estructura interna. Esa información es útil para resolver el problema de la subcadena en tiempo lineal, sea un texto *S* de longitud *m*, se pre-procesa (se construye el árbol) en tiempo *O(m)*.[20] Para buscar una subcadena P de longitud n basta con un tiempo de ejecución O(n).

### *3.2 Búsqueda aproximada permitiendo errores*

A) Algoritmos de Filtrado: Estos algoritmos realizan un filtro para descartar partes del texto en las que no es posible que el patrón se encuentre en forma aproximada. Si se desea buscar la palabra "abracadabra" en un texto de *n* caracteres y las palabras "abra", "cada" y "bra" aparecen como subpalabras sólo en los últimos *50* caracteres, se puede usar programación dinámica sobre los últimos *50+11* caracteres, y no sobre el texto completo. Esto se traduce en mayor velocidad de búsqueda. Los algoritmos de filtrado utilizan un método de búsqueda aproximada para encontrar las ocurrencias en todas aquellas porciones de texto que no fueron descartadas, cabe destacar también, que su eficiencia es válida sólo para valores de error k no superiores a cierto umbral. [24]

B) El algoritmo de Caracteres Lejanos: La idea básica es alinear el patrón con el texto y revisar el texto contando el número de caracteres lejanos, es decir, que no coinciden con el carácter del patrón con el que está alineado ni con cualquier carácter del patrón a distancia de *k* caracteres o menos. [16]

C) Algoritmo de q–gramas: Este algoritmo corresponde a la familia de los que se basan en buscar piezas del patrón en el texto en forma exacta. Un q -grama es una palabra de largo q. La idea básica es la siguiente, suponiendo que P (patrón) está a distancia menor o igual que k (error) de la sub-palabra Ti..j, y consideran una secuencia de por lo menos k operaciones, que transformen P en Ti..j. Cada operación altera por lo menos los q -gramas de P, por lo cual m − q + 1 − kq de los q -gramas del patrón deben aparecer en cualquier ocurrencia. Se denota m = m − q + 1 el número de q -gramas de P. El algoritmo que recorre cada ventana V de tamaño m − k (tamaño mínimo de una ocurrencia), contando el número de q -

gramas del patrón presentes en V. Existen dos posibilidades: si se encuentran menos de m − kq − 2k de los q -gramas de QP, entonces no hay ninguna ocurrencia q puede contener a la ventana. Ello porque el largo de una ocurrencia es a lo más m + k, por lo que como máximo 2k de los m q -gramas no han sido considerados en el conteo, en este caso la ventana es descartada por el filtro. Si se encuentran al menos m − kq − 2k de los q -gramas de QP, el filtro no descarta q la ventana y todos las ocurrencias que contienen a la ventana son encontradas con un algoritmo tradicional. Esto requiere revisar coincidencias en una ventana de texto de tamaño m + 3k, la ventana es verificada por el filtro. [21]

D) El algoritmo de Chang y Marr: Este algoritmo realiza una búsqueda aproximada de algunas subpalabras del texto en el patrón. Sean *R* y *S* dos palabras cualquiera. Se define *asm*(*R, S*) como la mínima distancia de la palabra *R* a las sub-palabras de *S* (Ecuación 2). Sea S una ocurrencia del patrón P en el texto T, se tiene que hay un conjunto total de coincidencias S de cantidad j.

$S = S^1, S^2, S^j$ Ecuación 1. Palabra S y sub-palabras a buscar $\sum_{i=1}^{j} \mathrm{asm}(S^i, P) \leq k$. Mínima distancia de la palabra R a las sub-palabras de S con el algoritmo Chang Marr. Para encontrar los resultados, el algoritmo divide el texto en ventanas disjuntas de largo v = (m − k)/2. Esta división garantiza que cada resultado contiene necesariamente a una de esas ventanas, cada una de estas es candidata a ser descartada por el filtro del algoritmo. Una ventana descartada significa que no existe ningún resultado conteniendo completamente esa ventana. Cuando el filtro no descarta una ventana, una porción de texto de tamaño (m + 3k)/2 debe ser verificada por un algoritmo tradicional. [22]

## IV. ALGORITMOS DE CLASIFICACIÓN DE DOCUMENTOS

En la ciencia de la RI, además de contar con un número de algoritmos de búsqueda, existen otros algoritmos encargados de la clasificación de los mismos. [23]. Dentro de estos algoritmos, los más comunes son: Algoritmo de Rocchio, Algoritmo del vecino más próximo y variantes, y algoritmos basados en redes neuronales.

### 4.1 Algoritmo de Rocchio

Este algoritmo conocido y aplicado en la realimentación de consultas. En este ámbito, la idea es sencilla: formulada y ejecutada una primera consulta, el usuario examina los documentos devueltos y determina cuáles le resultan relevantes y cuáles no. Con estos datos, el sistema genera automáticamente una nueva consulta, basándose en los documentos que el usuario señaló como relevantes o no relevantes. El algoritmo de Rocchio proporciona un sistema para construir el vector de la nueva consulta, recalculando los pesos de los términos de ésta y aplicando un coeficiente a los pesos de la consulta inicial, un coeficiente a los pesos de los documentos relevantes y otro coeficiente diferente a los de los no relevantes; esto con el fin de realizar una retroalimentación, y permitir que en una consulta futura, los resultados se acoplen mejor a la consulta realizada. [18] En categorización, el mismo algoritmo proporciona un sistema para construir los patrones de cada una de las categorías de documentos. Partiendo de una colección de entrenamiento, categorizada manualmente con anterioridad, y aplicando el modelo vectorial, se puede construir vectores patrón para cada una de las clases, considerando como ejemplos positivos los documentos de entrenamiento de esa categoría, y como ejemplos negativos los de las demás categorías.[13] Una vez que se tienen los patrones de cada una de las clases, el proceso de entrenamiento está concluido. Para categorizar nuevos documentos, se estima la similitud entre el nuevo documento y cada uno de los documentos ya clasificados; aquella comparación que genere un mayor índice de similitud indicará la categoría a la que se debe asignar ese documento.

### 4.2 Algoritmo del Vecino más cercano y variantes

El algoritmo del vecino más cercano (Nearest Neighbour, NN) es uno de los más sencillos de implementar. La idea principal es calcular la similitud entre el documento a clasificar y cada uno de los documentos de entrenamiento, teniendo en cuenta que estos documentos fueron clasificados manualmente, cuando se halla localizado el documento con mayor similitud ya se puede asignar una categoría al documento.

### 4.3 Redes Neuronales

Una red neuronal consta de varias capas de unidades de procesamiento o neuronas interconectadas; en el ámbito que nos ocupa la capa de entrada recibe términos, mientras que las unidades o neuronas de la capa de salida mapea clases o categorías. [14] Las interconexiones tienen pesos, es decir, un coeficiente que expresa la mayor o menor relevancia de la conexión. Es posible entrenar una red para que, dada una entrada determinada (los términos de un documento), produzca la salida deseada (la clase que corresponde a ese documento). El proceso de entrenamiento consta de un ajuste de los pesos de las interconexiones, a fin de que la salida sea la deseada. La idea de clasificación que manejan las redes neuronales corresponde a una serie de entradas (términos), que son analizados y mediante un cierto

número de procesos (capas en la neurona), se otorga una clasificación idónea(clase) al documento que se quiere clasificar. [11]

## V. TECNOLOGÍAS DE TEXTO

Entre estas tecnologías están aquellas que estudian el análisis morfológico, sintáctico y semántico del texto a fin de evaluar un posible algoritmo de búsqueda que satisfaga las necesidades planteadas inicialmente en el momento de construir un motor de búsqueda.

### 5.1 Analizadores morfológicos

El criterio que sigue el analizador para identificar las palabras es la identificación gráfica convencional de cadena de caracteres limitada por dos espacios en blanco, o limitada entre un espacio en blanco y un signo de puntuación. El usuario introduce una palabra y el software la descompone en los posibles morfemas que están presentes en la base de datos léxicos, luego determina qué análisis pueden ser correctos. [9] Para el idioma español existen varios sistemas de análisis morfológico, por ejemplo, MACO+ 0, FreeLing 0, FLANOM 0, el analizador morfológico integrado en MS Word, o el sistema AGME (Análisis y Generación Morfológica para el Español). El último, utiliza una metodología universal para los idiomas flexivos de desarrollo de los analizadores morfológicos (0, 0); se basa en la idea de "análisis a través de generación", es decir, primero se generan todas las posibles hipótesis para cada posible flexión en la palabra, incluyendo la flexión nula, y después se verifican las hipótesis generando las formas a partir de las posibles raíces. [10]

### 5.2 Analizadores Sintácticos

Se introduce la oración a analizar en el espacio habilitado para el mismo y el programa ejecuta la operación. Los analizadores sintácticos realizan tareas para combinación y relación de las palabras. Entre los analizadores sintácticos tenemos: el Analizador sintáctico del Grupo de Estructuras de Datos de la ULPGC, Analizador sintáctico Universidad de Barcelona, Analizador sintáctico Thera, Analizador sintáctico VISL. (Analizador multilenguaje), NeoBook Profesional Multimedia (educativo), APOLN Analizador Parcial de Oraciones en Lenguaje Natural, SUPAR. [11]

### 5.3 Desambiguadores

Cualquier forma a la que se puede asignar más de una interpretación ya sea morfosintáctica o semántica, es una entidad ambigua, ya que el programa deberá seleccionar la interpretación más adecuada teniendo en cuenta el contexto. Los corpus lingüísticos, constituyen la base para la constitución de un desambiguador. Se convierten en importantes repositorios de información convirtiéndose en la fuente principal de información para los sistemas y programas de desambiguación. Algunos ejemplos son: Longman Dictionary of Contemporary English (LDOCE), WordNet, WordNet Domains, EuroWordNet, SemCor (SEMantic COncoRdance), Senseval. [12]

## VI. CONCLUSIONES

Se evidenció que el trabajo arduo para la búsqueda de texto ha generado la inserción de varios algoritmos que resuelvan un problema de búsqueda de texto, no obstante, el pre-procesamiento de los datos puede ser un campo de investigación importante para hallar nuevas técnicas de clasificación, que permita al algoritmo que se elija como buscador, teniendo un tiempo de ejecución menor, disminuyendo el costo computacional y aumentado la efectividad.

Se plantea una combinación entre algoritmos que permita una búsqueda más sencilla y veloz. Se podría intentar combinar un algoritmo del modelo vectorial junto con el árbol de sufijos; hacer la búsqueda, no por palabras clave dentro del documento, sino, por sus sufijos, es posible que la complejidad aumente, pero es un coste bajo para la ganancia en tiempo de ejecución que esto generaría.

El algoritmo a utilizar en RI, en cualquier proyecto, depende del objetivo que se desea alcanzar, es decir; si lo que se pretende es realizar una búsqueda específica de un patrón la mejor opción será un algoritmo de búsqueda secuencial, sin embargo; si lo que se desea es un algoritmo que permita la búsqueda de relatividad con respecto a una consulta, lo idóneo es utilizar un algoritmo de búsqueda permitiendo errores.

El éxito de un algoritmo de búsqueda, está dado también por el proceso previo de recolección, análisis y clasificación de datos; esto incluye, si es necesario, el uso de un algoritmo de clasificación automática.

## VII. TRABAJOS FUTUROS

El trabajo a emprender debe implementar como mínimo los algoritmos anteriormente mencionados, con el fin de permitir, por medio del algoritmo de clasificación, una inserción de nuevos documentos y no limitar la búsqueda a los documentos existentes en la base de datos, y, un algoritmo vectorial, que permita una búsqueda aproximada de texto, encontrando similitud temática entre los documentos, mostrando como resultado un listado de documentos categorizados por medio de un ranking temático, es decir; se muestran los documentos con mayor relevancia entre la consulta hecha por el usuario y la concordancia existente con las palabras clave de cada documento.

## VIII. REFERENCIAS

*[1] C.D Manning, P Raghavan, y H. Schütze, Introduction to Information Retrieval. Cambridge University Press, 2008.*

*[2] Martínez Comeche Juan Antonio, Los modelos clásicos de Recuperación de información y su vigencia, Departamento de Biblioteconomía y documentación, Universidad Complutense de Madrid, http://eprints.ucm.es/5979/1/Modelos_RI_preprint.pdf.*

*[3] H. Tolosa Gabriel y R.A. Bordignon Fernando, Introducción a la Recuperación de Información, Universidad Nacional de Luján, Argentina*

*[4] De Lucca J. L, Scientific linguistic methodology employs theories and models as a basis for further analysis, cap 7: El Modelo del Espacio Vectorial.*

*[5] https://sites.google.com/site/busquedasecuencialdetexto/algoritmo-boyer-moore/algoritmoboyer-moore-horspool*

*[6]ict.udlap.mx/.../INDEXAMIENTO Y OPERACIONESCONTEXTO.ppt*

*[7] Ramírez Benavides M.Sc.KrysciaDaviana, Algoritmos de Búsqueda Secuencial de Texto, UCR –ECCI, CI-2414 Recuperación de Información*

*[8] http:// webdiis.unizar.es/ asignaturas/ TAP/ material/4.2.sufijos.pdf*

*[9] http://procesamiento-lenguaje-natural.webs.com/*

*[10] Gelbukh Alexander y Sidorov Grigori; Analizador Morfológico Disponible: un Recurso Importante para PLN en Español, Laboratorio de Lenguaje Natural y Procesamiento de Texto, Centro de Investigación en Computación(CIC), http://ciberia.ya.com/pgocegue/Paper/PR/Analizador%20Morfologico%20Disponible.pdf*

*[11] Ribera Xavier, Molina, Ferran Antonio Pla, Herramienta para el etiquetado léxico y análisis sintáctico de textos orientado a la construcción de corpus supervisados, Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y de Computación*

*[12] Tello Leal Edgar, La Desambiguación del Sentido de las Palabras: revisión metodológica. Revista No Solo Usabilidad journal - ISSN 1886-8592. email: info@nosolousabilidad.com, 17 de Abril de 2009.*

*[13] Téllez Valero Alberto, Extracción de Información con algoritmos de clasificación, Tonantzintla, Pue,2005.*

*[14] Dumais S., Platt J., Heckerman D., and Sahami M., Inductive learning algorithms and representations for text categorization. In Proceedings of the Seventh International Conference on Information Retrieval and Knowledge Management (ACM-CIKM'98), pp. 148-155, 1998.*

*[15] Casasola Murillo Edgar; Recuperación de información en el contexto de la ciencia de la computación, Volumen 6; Septiembre 23, 2006*

*[16] Telha Cornejo Claudio Andrés, Búsqueda Aproximada permitiendo errores, tesis para optar al grado de magíster en ciencias, mención computación, memoria para optar al título de ingeniero civil en computación, memoria para optar al título de ingeniero civil matemático, Santiago de Chile; Agosto de 2007*

*[17] Montes y Gómez Manuel, Minería de texto: Un nuevo reto computacional, Centro de investigación en computación*

*[18] Pérez Abelleira M. Alicia y Cardoso Carolina A, Minería de texto para la categorización automática de documentos*

*[19] Apache UIMA Development Community, 2008, UIMA Tutorial and Developers' Guides, version 2.2, http://incubator.apache.org/uima/downloads/releaseDocs/2.2.2-incubating/docs/html/.*

*[20] Abeliuk Kimelman Andrés Jonathan, Árboles de sufijos comprimidos para textos altamente repetitivos, Universidad de Chile Facultad de ciencias fisicas y matemáticas, Departamento de ciencias de la computación.*

*[21] Thierry Lecroq Christian Charras,. HandBook of Extract String-Matching Algorithms.*

*[22] Sierra Sánchez María Rita, Mejora de algoritmos de búsqueda Heurística mediante poda por dominancia, Aplicación a problemas de Scheduling.*

*[23] Figuerola Carlos G., Alonso Berrocal José L.; Zazo Rodríguez Ángel F., Rodríguez Emilio, Algunas Técnicas de Clasificación Automática de Documentos, Universidad de Salamanca.*

*[24] Avendaño Carlos, Feregrino Claudia, Navarro Gonzalo, Mejorando un Algoritmo para Búsqueda Aproximada, Departamento de Ciencias de la Computación, Universidad de Chile; Santiago, Chile.*

*[25] Zazo Rodríguez Ángel F., Figuerola Paniagua Carlos G., Alonso Berrocal José Luis, Gómez Díaz Raquel; Recuperación de información utilizando el modelo vectorial. Participación en el taller CLEF−2001. Departamento de Informática y Automática. Universidad de Salamanca.*